Terms used **document** **folders** and **tasks**

Found **732** of **198,146**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown
Relevance scale

1 Coordination models, languages and applications (CM): Orchestrating document-based workflows with X-Folders
Davide Rossi
March 2004 **Proceedings of the 2004 ACM symposium on Applied computing SAC '04**
Publisher: ACM Press
Full text available: pdf(198.06 KB) Additional Information: full citation, abstract, references

This paper introduces X-Folders: a software environment for multi-party document-based processes that aims at supporting the implementation of workflow processes involving multiple users that interact by means of documents stored in special, reactive, folders. When the documents inside a folder reach a given state a task is triggered, tasks orchestrate a set of web services in order to enact the workflow. X-Folders is based on a model that promotes the implementation of applications distributed ...

Keywords: CSCW, coordination, orchestration, peer-to-peer, workflows

2 Genre, task, topic and time: facets of personal digital document management
Sarah Henderson
July 2005 **Proceedings of the 6th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction: making CHI natural CHINZ '05**
Publisher: ACM Press
Full text available: pdf(175.54 KB) Additional Information: full citation, abstract, references, index terms

Most operating systems provide the ability to create folders to contain documents, and to nest these folders to create a hierarchical organization. However, little is known about the kinds of folders people create using this type of organizing scheme, or how they structure those folders.Exploratory research was conducted, analyzing the folder structures of six knowledge workers and it was found that most folder names represent the genre, task, topic or time dimension of the documents they contai ...

Keywords: document organization, personal digital document management, personal information management

3 OTM: specifying office tasks
F. H. Lochovsky, J. S. Hogg, A. O. Mendelzon, S. P. Weiser

April 1988 **ACM SIGOIS Bulletin , Proceedings of the ACM SIGOIS and IEEECS TC-OA 1988 conference on Office information systems**, Volume 9 Issue 2-3
Publisher: ACM Press

Full text available: pdf(988.62 KB) Additional Information: full citation, abstract, references, citings, index terms

While there are many difficulties in computerizing office tasks, two of the major ones are a lack of appropriate end-user facilities for specifying office tasks and inadequate system-level support for managing office tasks. We are investigating these two issues within the Office Task Manager (OTM) project at the University of Toronto. To address the user-level aspects of specifying office tasks, we believe that a programming-by-example approach to office task specification holds much promis ...

4 TeleNotes: managing lightweight interactions in the desktop
Steve Whittaker, Jerry Swanson, Jakov Kucan, Candy Sidner
June 1997 **ACM Transactions on Computer-Human Interaction (TOCHI)**, Volume 4 Issue 2
Publisher: ACM Press

Full text available: pdf(1.01 MB) Additional Information: full citation, abstract, references, citings, index terms

Communication theories and technology have tended to focus on extended, formal meetings and have neglected a prevalent and vital form of workplace communication—namely, lightweight communication. Unlike formal, extended meetings, lightweight interaction is brief, informal, unplanned, and intermittent. We analyze naturalistic data from a study of work-place communication and derive five design criteria for lightweight interaction systems. These criteria require that systems for lightwe ...

Keywords: audio, awareness, computer-media spaces, conversation management, impromptu communication, informal communication, interpersonal communications, lightweight interaction, mediated communication, remote collaboration, task management, video

5 Document interaction: Documents at Hand: Learning from Paper to Improve Digital Technologies
Olha Bondarenko, Ruud Janssen
April 2005 **Proceedings of the SIGCHI conference on Human factors in computing systems CHI '05**
Publisher: ACM Press

Full text available: pdf(375.17 KB) Additional Information: full citation, abstract, references, citings, index terms

In this paper the results of a two-year ethnographic study of the personal document management of 28 information workers is described. Both the paper *and* digital domain were taken into account during the study. The results reaffirmed that document management is strongly related to task management. Digital tools do not adequately support two important user needs related to task management, namely that documents should be embedded within meaningful (task-related) context information, and th ...

Keywords: document management, email, ethnography, paper, paper-digital integration

6 Merging multiple perspectives in groupware use: intra- and intergroup conventions
Gloria Mark
November 1997 **Proceedings of the international ACM SIGGROUP conference on Supporting group work: the integration challenge GROUP '97**
Publisher: ACM Press

Full text available: pdf(1.52 MB) Additional Information: full citation, references, citings, index terms

Keywords: CSCW, conventions, groupware, intergroup, shared workspace

7 Understanding the role of documents in a hierarchical flow of work
Peter Mambrey, Mike Robinson
November 1997 **Proceedings of the international ACM SIGGROUP conference on Supporting group work: the integration challenge GROUP '97**
Publisher: ACM Press
Full text available: pdf(1.27 MB) Additional Information: full citation, references, citings, index terms

Keywords: CSCW, boundary object, organisational processes, work practice, workflow

8 Industrial session: Task-based process know-how reuse and proactive information delivery in TaskNavigator
Harald Holz, Oleg Rostanin, Andreas Dengel, Takeshi Suzuki, Kaoru Maeda, Katsumi Kanasaki
November 2006 **Proceedings of the 15th ACM international conference on Information and knowledge management CIKM '06**
Publisher: ACM Press
Full text available: pdf(1.24 MB) Additional Information: full citation, abstract, references, index terms

Knowledge management approaches for weakly-structured, adhoc knowledge work processes need to be lightweight, i.e., they cannot rely on high upfront modeling efforts. This paper presents TaskNavigator, a novel prototype to support weakly-structured processes by integrating a standard task list application with a state-of-the-art document classification system. The resulting system allows for a task-oriented view on office workers' personal knowledge spaces in order to realize a proactive and con ...

Keywords: agile workflows, proactive information delivery, process-oriented knowledge management

9 Enhancing workflows by web technology
Wolfgang Gräther, Wolfgang Prinz, Sabine Kolvenbach
November 1997 **Proceedings of the international ACM SIGGROUP conference on Supporting group work: the integration challenge GROUP '97**
Publisher: ACM Press
Full text available: pdf(1.34 MB) Additional Information: full citation, references, citings, index terms

Keywords: HTML, Internet, electronic circulation folder, workflow

10 Support of cooperative work by electronic circulation folders
B. Karbe, N. Ramsperger, P. Weiss
March 1990 **ACM SIGOIS Bulletin , Proceedings of the ACM SIGOIS and IEEE CS TC-OA conference on Office information systems**, Volume 11 Issue 2-3
Publisher: ACM Press
Additional Information: full citation, abstract, references, citings, index

Full text available: pdf(898.86 KB) terms

ProMInanD's migration system deals with cooperative office work on different types of office tasks which consist of steps to be carried out by person playing office roles. This typical kind of cooperation is supported by ProMInanD's Electronic Circulation Folders (ECF). Instantiations of task related types of ECF's migrate automatically through an office organization. In the contents of an ECF the office workers' contribution to a task are kept as arbitrary kinds of documents. The migration ...

11 Integrating tools and tasks: UMEA: translating interaction histories into project contexts
Victor Kaptelinin
April 2003 **Proceedings of the SIGCHI conference on Human factors in computing systems CHI '03**
Publisher: ACM Press
Full text available: pdf(232.45 KB) Additional Information: full citation, abstract, references, citings, index terms

Virtual environments based on the desktop metaphor provide limited support for creating and managing project-specific work contexts. The paper discusses existing approaches to supporting higher-level user activities and presents a system named UMEA (User-Monitoring Environment for Activities). The design of the system is informed by activity theory. The system: (a) organizes resources into project-related pools consisting of documents, folders, URLs, and contacts, (b) monitors user activities, (...

Keywords: activity theory, interaction histories

12 Influence of exception handling on the support of cooperative office work
Bernhard H. Karbe, Norbert G. Ramsperger
December 1990 **ACM SIGOIS Bulletin**, Volume 11 Issue 4
Publisher: ACM Press
Full text available: pdf(1.11 MB) Additional Information: full citation, abstract, citings, index terms

Cooperative work on office tasks can be understood as consisting of steps to be carried out by persons acting in organizational functions. In many cases, neither the number of steps nor the persons involved are known in advance. Often, deviations from predefined cooperation procedures and exception handling become necessary. Moreover, many tasks are completely unformalized. Thus, support of exception handling has become a crucial requirement. ProMInanD represents office tasks by means of Electro ...

13 Application-oriented usage quality: the tools and materials approach
Carola Lilienthal, Heinz Züllighoven
November 1997 **interactions**, Volume 4 Issue 6
Publisher: ACM Press
Full text available: pdf(1.07 MB) Additional Information: full citation, references, citings, index terms, review

14 Knowledge-based document retrieval in office environments: the Kabiria system
Augusto Celentano, Maria Grazia Fugini, Silvano Pozzi
July 1995 **ACM Transactions on Information Systems (TOIS)**, Volume 13 Issue 3
Publisher: ACM Press
Full text available: pdf(2.14 MB) Additional Information: full citation, abstract, references, citings, index terms, review

In the office environment, the retrieval of documents is performed using the concepts contained in the documents, information about the procedural context where the documents are used, and information about the regulations and laws that discipline the life of documents within a given application domain. To fulfill the requirements of such a sophisticated retrieval, we propose a document retrieval model and system based on the representation of knowledge describing the semantic contents of d ...

Keywords: browser, class, hypertext, instance, knowledge base, link, object orientation, user interface

15 Moksha: exploring ubiquity in event filtration-control at the multi-user desktop
Rameshsharma Ramloll, John A. Mariani
March 1999 **ACM SIGSOFT Software Engineering Notes , Proceedings of the international joint conference on Work activities coordination and collaboration WACC '99**, Volume 24 Issue 2
Publisher: ACM Press
Full text available: pdf(1.64 MB) Additional Information: full citation, abstract, references, citings, index terms

Collaborative systems need to provide some means for users to be aware of peer activities. Common approaches involve broadcasting events generated as a result of a particular user's actions at the interface to others. Rather than flooding users with information about all activities occurring in the shared environment, filtration techniques allow each user to be exposed to relevant awareness information. Such techniques are often based on user configurable agents. Unfortunately, these so far do n ...

Keywords: auditory display, awareness, common information space, multi-users desktop system, multimedia browsing

16 2b---Hypertext Systems: Organizing shared enterprise workspaces using component-based cooperative hypermedia
Jessica Rubart, Jörg M. Haake, Daniel A. Tietze, Weigang Wang
September 2001 **Proceedings of the twelfth ACM conference on Hypertext and Hypermedia HYPERTEXT '01**
Publisher: ACM Press
Full text available: pdf(380.50 KB) Additional Information: full citation, abstract, references, citings, index terms

Cooperative work in Extended Enterprises needs a flexible shared workspace for team members to access and manipulate shared information objects in a well-coordinated working process. Current shared workspace systems do not adequately support the evolving character of shared workspaces as needed by Extended Enterprises, i.e. the dynamic cooperation processes, various kinds of shared information contents and the set of cooperative tools. In this paper, the usage scenarios and requirements devel ...

17 Support for workflows in a ministerial environment
Wolfgang Prinz, Sabine Kolvenbach
November 1996 **Proceedings of the 1996 ACM conference on Computer supported cooperative work CSCW '96**
Publisher: ACM Press
Full text available: pdf(1.44 MB) Additional Information: full citation, references, citings, index terms

Keywords: digital signatures, electronic circulation folder, participatory design, shared

workspaces, workflow

18 Using collaborative filtering to weave an information tapestry
David Goldberg, David Nichols, Brian M. Oki, Douglas Terry
December 1992 **Communications of the ACM**, Volume 35 Issue 12
Publisher: ACM Press
Full text available: pdf(3.12 MB) Additional Information: full citation, references, citings, index terms, review

Keywords: information filtering, tapestry

19 Presto: an experimental architecture for fluid interactive document spaces
Paul Dourish, W. Keith Edwards, Anthony LaMarca, Michael Salisbury
June 1999 **ACM Transactions on Computer-Human Interaction (TOCHI)**, Volume 6 Issue 2
Publisher: ACM Press
Full text available: pdf(409.04 KB) Additional Information: full citation, abstract, references, citings, index terms

Traditional document systems use hierarchical filing structures as the basis for organizing, storing and retrieving documents. However, this structure is very limited in comparison with the rich and varied forms of document interaction and category management in everyday document use. Presto is a prototype document management system providing rich interaction with documents through meaningful, user-level document attributes, such as "Word file," "published paper," &l ...

Keywords: attribute/value systems, direct manipulation, document management

20 Using knowledge to predict and manage: Leveraging characteristics of task structure to predict the cost of interruption
Shamsi T. Iqbal, Brian P. Bailey
April 2006 **Proceedings of the SIGCHI conference on Human Factors in computing systems CHI '06**
Publisher: ACM Press
Full text available: pdf(1.64 MB) Additional Information: full citation, abstract, references, index terms

A challenge in building interruption reasoning systems is to compute an accurate cost of interruption (COI). Prior work has used interface events and other cues to predict COI, but ignore characteristics related to the structure of a task. This work investigates how well characteristics of task structure can predict COI, as objectively measured by resumption lag. In an experiment, users were interrupted during task execution at various boundaries to collect a large sample of resumption lag value ...

Keywords: attention, interruption, learning, task models, workload

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Object Lens: A "Spreadsheet" for Cooperative Work

KUM-YEW LAI, THOMAS W. MALONE, and KEH-CHIANG YU
Massachusetts Institute of Technology

Object Lens allows unsophisticated computer users to create their own cooperative work applications using a set of simple, but powerful, building blocks. By defining and modifying templates for various semistructured objects, users can represent information about people, tasks, products, messages, and many other kinds of information in a form that can be processed intelligently by both people and their computers. By collecting these objects in customizable folders, users can create their own displays which summarize selected information from the objects in table or tree formats. Finally, by creating semiautonomous agents, users can specify rules for automatically processing this information in different ways at different times.

The combination of these primitives provides a single consistent interface that integrates facilities for object-oriented databases, hypertext, electronic messaging, and rule-based intelligent agents. To illustrate the power of this combined approach, we describe several simple examples of applications (such as task tracking, intelligent message routing, and database retrieval) that we have developed in this framework.

Categories and Subject Descriptors: H.1.2 [**Models and Principles**]: User/Machine Systems; H.2.1 [**Database Management**]: Logical Design—*data models*; *schema and subschema*; H.2.3 [**Database Management**]: Languages—*data description languages* (*DDL*); H.2.4 [**Database Management**]: Systems—*distributed systems*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.4 [**Information Storage and Retrieval**]: Systems and Software; H.4.1 [**Information Systems Applications**]: Office Automation; H.4.3 [**Information Systems Applications**]: Communications Applications; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*office automation*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*frames and scripts*; *representations*; I.7.2 [**Text Processing**]: Document Preparation—*format and notation*

General Terms: Design, Economics, Human Factors, Management

Additional Key Words and Phrases: Computer-supported cooperative work, hypertext, information lens, intelligent agents, object-oriented databases, semiformal systems

1. INTRODUCTION

It is common in the computer industry today to talk about the "next spreadsheet"—to claim that a particular application will be the "next spreadsheet" or to wonder what the "next spreadsheet" will be (e.g., [7]). Usually the term

The work described in this paper was supported, in part, by Wang Laboratories, Xerox Corporation, General Motors/Electronic Data Systems, Bankers Trust Company, the Development Bank of Singapore, and the Management in the 1990s Research Program at the Sloan School of Management, MIT.
Authors' current addresses: K.-Y. Lai, 10-174 Block 129, Bukit Merah View, Singapore 0315, Republic of Singapore; T. W. Malone and K.-C. Yu, Sloan School of Management (E53-333), Massachusetts Institute of Technology, Cambridge, Mass. 02139.

"spreadsheet" is used in this context simply to connote a product that embodies some kind of design breakthrough and is very successful.

We will focus here on a more specific property of spreadsheet programs: They make a restricted, but nevertheless, very flexible and useful, set of computational capabilities extremely easy to use. It is, of course, possible to do any computation a spreadsheet can do in a general purpose programming language. But because doing these things with a spreadsheet program is so much more convenient, the number of people who can use computers to do them increases by orders of magnitude.

In this paper, we describe an early prototype of a system, called Object Lens, which we believe shares this property of spreadsheets: It makes accessible to unsophisticated computer users a set of computational and communications capabilities that, although limited, are quite flexible and useful for supporting a wide variety of cooperative work activities. In other words, we use the term "spreadsheet" here, not to connote financial modeling or constraint languages, but to connote a flexible infrastructure in which people who are not professional programmers can create or modify their own computer applications.

In the remainder of this paper we (1) describe the key ideas used in the design of Object Lens, (2) show how these ideas are realized in Object Lens features, and (3) illustrate the flexibility and usefulness of these ideas in several examples of cooperative work.

1.1 Three Views of Object Lens

Before proceeding, it is useful to point out three ways of viewing the Object Lens system.

(1) *Object Lens is the "second generation" of the Information Lens system.* Object Lens is based on our experience with using and enhancing the Information Lens [13, 14], an intelligent system for information sharing and coordination. A very large number of the enhancements that we and others have suggested for the Information Lens are included in Object Lens. Like the Information Lens, Object Lens uses ideas from artificial intelligence and user interface design to represent knowledge in such a way that both people and their computational agents can process it intelligently. Object Lens, however, is a significant generalization of the Information Lens. It potentially goes far beyond the Information Lens in the kinds of knowledge that can be represented and the ways that information can be manipulated.

(2) *Object Lens is a user interface that integrates hypertext, object-oriented databases, electronic messaging, and rule-based intelligent agents.* Object Lens does not include all the capabilities of all these different classes of systems, but we have been surprised at how cleanly a large portion of these diverse capabilities can be integrated. The key contribution of Object Lens is thus not the completeness of its implementation, but the integration of its user interface. Since the capabilities of these different kinds of systems are no longer separate applications, each capability is more useful than it would be alone, and the resulting system is unusually flexible.

(3) *Object Lens is a knowledge-based environment for developing cooperative work applications.* In the original Information Lens system, we developed specific

applications for information sharing, meeting scheduling, project management, and computer conferencing. From the viewpoint of knowledge-based systems, these applications only included knowledge about different types of messages: the kinds of information the messages contained and the kinds of actions they could evoke. Object Lens, by contrast, can include explicit knowledge about many other kinds of objects such as people, tasks, meetings, products, and companies. We expect that the flexible tools Object Lens provides for dealing with these diverse kinds of knowledge will significantly increase the ease of developing a much wider range of applications. This last view of Object Lens, which emphasizes its flexibility, is our primary focus in this paper.

2. KEY IDEAS

One of the most important characteristics of Object Lens is that it is a *semiformal system.* We define a semiformal system as a computer system that has the following three properties: (1) it represents and automatically processes certain information in formally specified ways; (2) it represents and makes it easy for humans to process the same or other information in ways that are not formally specified; and (3) it allows the boundary between formal processing by computers and informal processing by people to be easily changed.

Semiformal systems are most useful when we understand enough to formalize in a computer system some, but not all, of the knowledge relevant to acting in a given situation. Such systems are often useful in supporting individual work, and we believe they are especially important in supporting cooperative work where there are usually some well-understood patterns in people's behavior and also a very large amount of other knowledge that is potentially relevant but difficult to specify.

In order to create such a flexible semiformal system, the knowledge embodied in the system must be exposed to users in a way that is both *visible* and *changeable* (cf., [20]). That is, users must be able to easily see and change the information and the processing rules included in the system. In Object Lens, there are three key ideas about how to represent and expose knowledge to users:

(1) "Passive" information is represented in *semistructured objects* with template-based interfaces;
(2) "Aggregate" information from collections of objects is summarized in *customizable folders*; and
(3) "Active" rules for processing information are represented in *semiautonomous agents.*

In the remainder of Section 2, we provide an overview of how these three components allow us to expose knowledge to users in a way that is both visible and changeable. Detailed descriptions of the system features are in Section 3.

2.1 Semistructured Objects

Users of the Object Lens system can create, modify, retrieve, and display objects that represent many physically or conceptually familiar things such as messages, people, meetings, tasks, manufactured parts, and software bugs. The system provides an interface to an object-oriented database in the sense that (1) each

object includes a collection of fields and field values, (2) each object type has a set of actions that can be performed upon it, and (3) the objects are arranged in a hierarchy of increasingly specialized types with each object type "inheriting" fields, actions, and other properties from its "parents" [4, 16, 17]. For example, a `TASK` object may have fields like `Requestor`, `Performer`, `Description`, and `Deadline`; a `PERSON` object may have fields like `Name`, `Phone`, `Address`, and `Job title`; and a `STUDENT` object may add fields like `Year` and `Advisor` to the fields present in all `PERSON` objects. Some objects (e.g., `MESSAGES`) have specialized actions defined for them (e.g., `Answer` and `Forward`). As described in more detail below, we have provided rudimentary facilities for saving and sharing objects, and we are currently exploring ways to link our interface to remote databases.

The objects in Object Lens, like messages in the Information Lens, are *semistructured* in the sense that users can fill in as much or as little information in different fields as they desire, and the information in a field is not necessarily of any specific type (e.g., it may be free text such as "I don't know").

2.1.1 *Template-Based User Interfaces.* Users can see and change objects through a particularly natural form of template-based user interface. These interfaces have a number of virtues. For instance: (1) they resemble forms, with which users are already familiar; (2) they conveniently inform users about the fields contained in an object and about other information such as the likely alternatives for different fields; and (3) their use is consistent across many different kinds of objects. We discuss later how this interface approach, which was used for messages and rules in the Information Lens, can be easily generalized to many different kinds of objects.

2.1.2 *Relationships Among Objects.* Users can easily see and change the relationships among objects by inserting and deleting *links* between the objects. For instance, the `Requestor` and `Performer` fields of a `Task` object might contain links to the `Person` objects that represent, respectively, the person who requested that the task be done and the person who performs the task. Then, for instance, when the user looks at the `Task` object, it is easy to get more information (e.g., the phone numbers) about the people involved with the task. We discuss later how this capability of linking objects to each other provides a rudimentary *hypertext* system as a special case (see [1] for an extensive review of hypertext systems). We also show how it is also possible for an object, to which a link appears, to be displayed as an *embedded template* inside the original template.

2.1.3 *Tailorable Display Formats.* Users have several options for changing the ways they see objects. For instance, they can easily (1) select which fields are to be shown and which are to be suppressed, (2) rename selected fields, and (3) specify the default and alternative values the system presents for individual fields.

2.1.4 *Inheritance Hierarchy for Objects.* The creation and modification of type definitions is simplified by arranging object types in an inheritance hierarchy (e.g., [17]). New types of objects are defined as specializations of existing object types, and they automatically "inherit" all properties of the existing objects

except those which are specifically "overridden." Since most of the information about new object types can thus be "inherited" from existing types, rather than having to be reentered each time, creating new object types becomes simpler. Also, when an object type definition is changed later, the changes are automatically "inherited" by the specializations of that object type.

2.2 Customizable Folders

Users of Object Lens can group collections of objects together into special kinds of objects called `Folders`. For instance, folders can be created for groups of people (e.g., project teams, company directory), tasks (e.g., those completed, those to be done by you, those to be done by others), messages (grouped according to topic or urgency), and so forth. Users can also easily customize their own displays to summarize the contents of objects in a folder. For instance, they can select certain fields to be displayed in a *table* with each row representing an object in the folder and each column representing a field. They can also select fields from which the links between objects can be used to create a *tree* (or graph) display with each object represented as a node in the tree and each link in the selected field represented as a line between nodes.

2.3 Semiautonomous Agents

Users of the Object Lens system can create rule-based "agents" that process information automatically on behalf of their users (see [2] for an extended discussion of agents). These agents provide a natural way of partitioning the tasks performed automatically by the system. As discussed later, agents can be "triggered" by events such as the arrival of new mail, the appearance of a new object in a specified folder, the arrival of a prespecified time, or an explicit selection by the user. When an agent is triggered, it applies a set of rules to a specified collection of objects. If an object satisfies the criteria specified in a rule, the rule performs some prespecified action. These actions can be general actions such as retrieving, classifying, mailing, and deleting objects or object-specific actions such as loading files or adding events to a calendar.

The agents in Object Lens are *autonomous* in the sense that once they have been created, they can take actions without the explicit attention of a human user. They are only *semiautonomous*, however, in the sense that (a) they are always controlled by a human user (that is, all their rules can be easily seen and changed by their human user), and (b) they may often "refer" objects to their human user for action (e.g., by leaving the object in the user's inbox) instead of taking any actions on their own.

2.3.1 *Descriptions.* Since agents and rules are themselves objects, users can see and modify them with the same template-based user interface that is used for all other kinds of objects. To specify the criteria for when rules should act upon a given object, users create *descriptions* of the objects to which the rules apply. A description is simply a partially filled-in template for an object of a particular type. Descriptions can also include *embedded descriptions* that specify characteristics that must be satisfied by objects to which the original object is linked. For instance, a description of a Task might include an embedded description of the person who performs the task. These embedded descriptions (like

those in the Rabbit system [19]), allow users to easily specify object retrieval operations that are equivalent to "joins" followed by "selects" in a relational database.

3. SYSTEM FEATURES

In this section, we describe in more detail the basic system features of Object Lens and illustrate them with simple examples (see [10] for more details about an earlier version of the system). The Object Lens system is implemented in Interlisp-D on Xerox 1100 series workstations connected by an Ethernet. The system makes heavy use of the object-oriented programming environment provided by Loops and the built-in text editor, Tedit. Except where otherwise noted, everything described here has been implemented, but many features have not yet been extensively tested. As of this writing, the basic mail handling capabilities have been used regularly by two people in our development group for about six months, and the other facilities have received limited testing.

3.1 Terminology: Objects and Templates

Before proceeding it is helpful to clarify some terminology concerning objects and templates. First, we distinguish between *object types* (or "classes") and specific *object instances* (e.g., see [5]). We use the term *object type* to refer to a kind of object (such as Person or Task) and the term *object instance* (or simply "instance") to refer to a specific example of one of these object types (e.g., "Joe Smith" or "Task No. 17"). In contexts in which the distinction between object types and object instances is not critical, we use the term *objects* to include both.

We also use the term *template* in two ways. First, in a general sense, we use the term *template* to mean any semistructured collection of fields and field contents. Most of a user's interactions with Object Lens are based on such templates. Second, in the Object Lens screen displays, we use the word `Template` to mean object type definition. (When we use `Template` in this specialized sense, we always capitalize it.) For instance, users can change the display format for all `Person` objects by editing the `Template` that defines the `Person` object type.

3.2 Editing Instances

Figure 1 shows a template for an instance of a `Person`. Using the built-in text editor, users can insert text or bitmaps in any field. In addition, when users click on a field name with the mouse, a list of likely alternative values for that field appears in a pop-up menu. The alternatives may be links to other objects or just text strings. Selecting one of these alternatives causes the alternative to be automatically inserted in the field. For instance, the figure contains a link to the `Person` object representing Kum-Yew Lai's supervisor. To insert links to objects that are not in the alternatives list, the user (a) positions the cursor at the place in the template where the link is to be inserted, (b) selects the `Add Link` option from the menu at the top of the window, and then (c) points to the object to which the link should be made. After a link is inserted, clicking on it with the mouse causes the object it points to to appear on the screen.

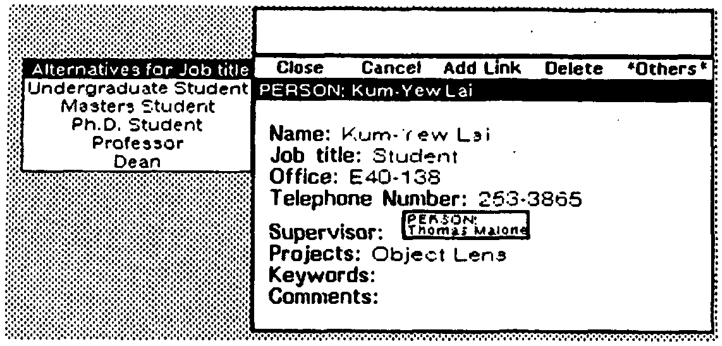

Fig. 1. Objects can be edited with a simple template editor. Fields can include text, graphics, or links to other objects.

In the current version of Object Lens, users can insert any combination of text, numbers, links, and bitmaps in any field. Then, in some cases, type checking is done when the editing window for the instance is closed or when certain kinds of processing are done. For instance, the `To` and `cc` fields are checked for valid addresses before sending messages and the "move to" field in rule actions is checked for valid folders (see Sections 3.5 and 3.7 for descriptions of rules and folders). In future versions of Object Lens, we may experiment with more restrictive type enforcement in certain fields. For instance, it should probably be impossible to even insert something other than a folder in the "move to" field of a rule action.

Figure 2 shows a slightly more complex template; this one is for a `Bug Fix Request` message. One of the fields of this template is the `Bug` to be fixed, and the value of this field is a link to a `Bug` object. In this case, instead of simply showing a link to the `Bug` object, the template contains an *embedded template* for the `Bug` object itself. The fields in this embedded template can be edited just like the rest of the fields in the template. We later discuss how users can specify whether links to other objects should be displayed as *link icons* (as in Figure 1) or as *embedded templates* (as in Figure 2).

3.3 Creating New Instances

To create and display a new instance of an object type that already exists, users click with the mouse on the definition (i.e., the `Template`) for that object type. Figure 3 shows the `Templates` currently included in our system. For instance, to send a new message, users click on the `Template` for the type of message they want to create; to create a new person object, users click on the `Person Template`. Then an object instance, like those shown in Figures 1 and 2, appear, and the user can fill it in.

3.4 Creating New Object Types

To create a new object type, users click (with both mouse buttons instead of the left one) on the `Template` for the "parent" object type (see Figure 3). This causes a menu to appear showing alternative actions that can be performed on a `Template`. One of these actions is to `Create a subtemplate`. When the

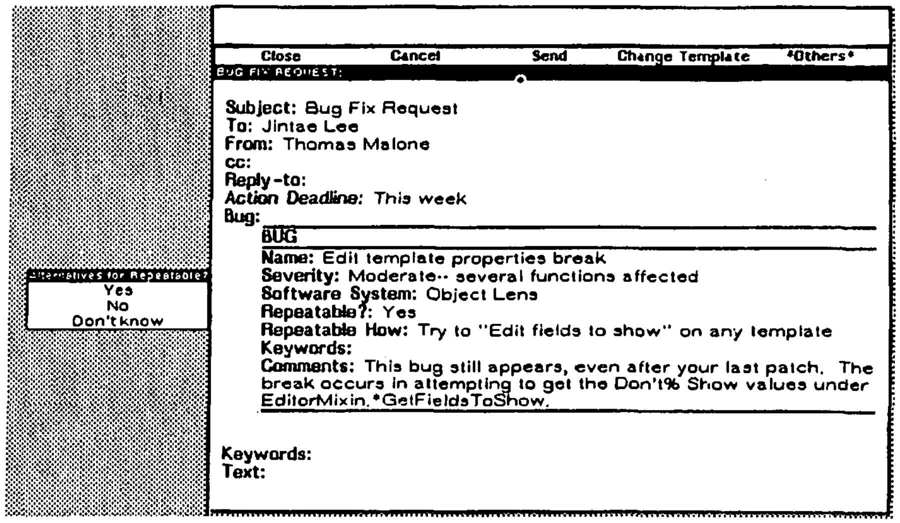

Fig. 2. Embedded templates allow related objects to be viewed and edited simultaneously.

user selects this action, a new `Template` is created with all the fields and properties of its "parent." Then users can add fields to the new Template or change its display format and other properties.

In the current version of Object Lens, all `Things` have three fields: `Name`, `Keywords`, and `Comments`. All objects inherit these fields, though as discussed below, some objects rename these fields or suppress their display. For instance, `Messages`, rename the `Name` field to be `Subject` and the `Comments` field to be `Text`.

3.5 Changing the Display Format and Other Properties of Object Types

To change the display format or other properties of an object type, users "edit" the `Template` that defines the object type. Users make these changes by selecting actions from the menu that appears when they click on the `Template` (as shown in Figure 3) with both mouse buttons. In this way, users can change (a) which fields of the objects are actually displayed, (b) the names of the fields that are displayed, (c) the alternative values that are displayed for each field, (d) the default values that are displayed in each field when new instances are created, and (e) whether the links in a field should be shown as link icons (see Figure 1) or as embedded templates (see Figure 2). In this mode, users can also add or delete fields from a template. All the changes made to a template are applied to old instances of an object type as well as to newly created ones. For example, if a user changes the name of a field, then the new name is shown when any old instances are redisplayed.

We anticipate that this system will be used with a core set of object types shared by the users in a group, and that the fields in these types will be modified only by an "authorized view administrator." Other users will be able to change the display format of these types (e.g., suppress the display of a field or change its name), but they would not be able to delete or add fields to these "official"

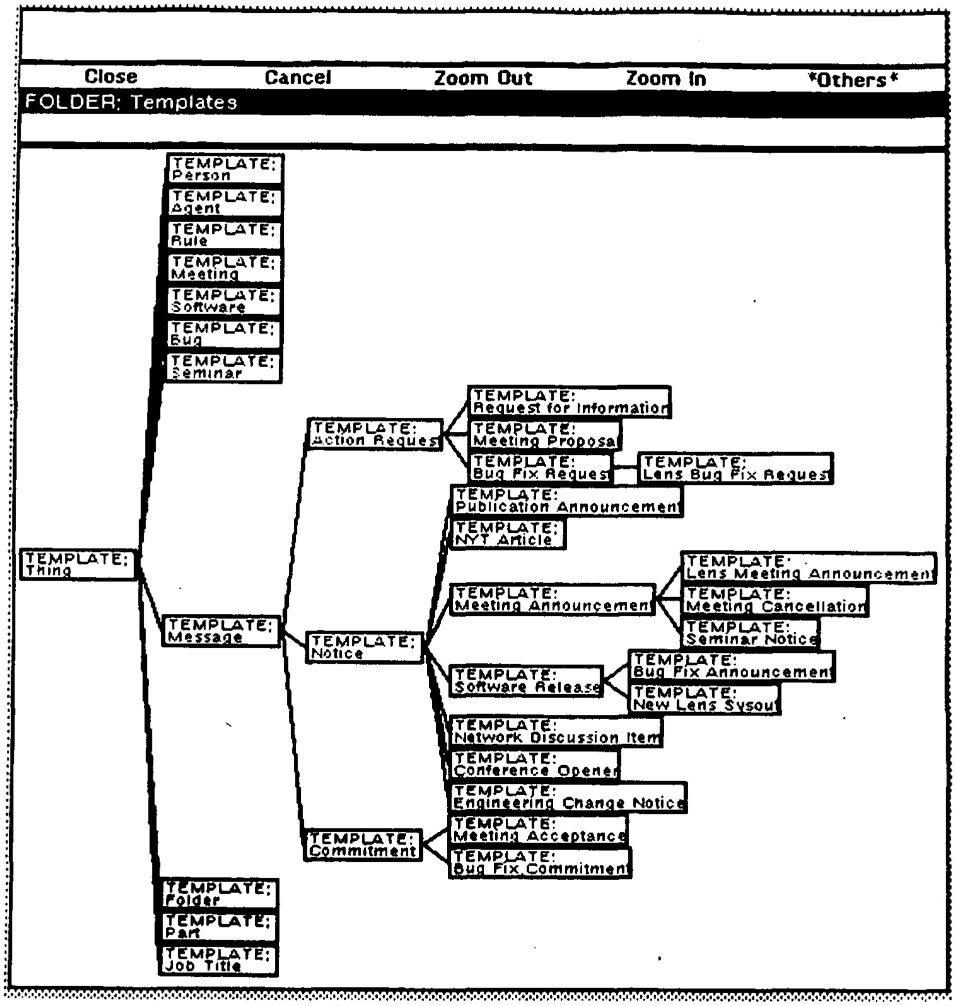

Fig. 3. Object types are defined by a set of `Templates`.

types. All users would, however, be able to create their own types as specializations of the official types, and for these types they could add and delete new fields as desired. Elsewhere [11, 12], we have proposed a scheme for letting an arbitrarily large number of groups share partially overlapping sets of type definitions in arbitrary ways. One of the key ideas of this scheme is that specialized types created by one group can be interpreted by members of another group as instances of the most specific "ancestor" type that both groups share. For instance, a "Student" object created by one group might be interpreted as a "Person" object by another group that does not have a definition for "Student."

3.6 Folders

As previously noted, Object Lens users can group collections of objects together into special kinds of objects called `Folders` (see Figure 4). An object can be

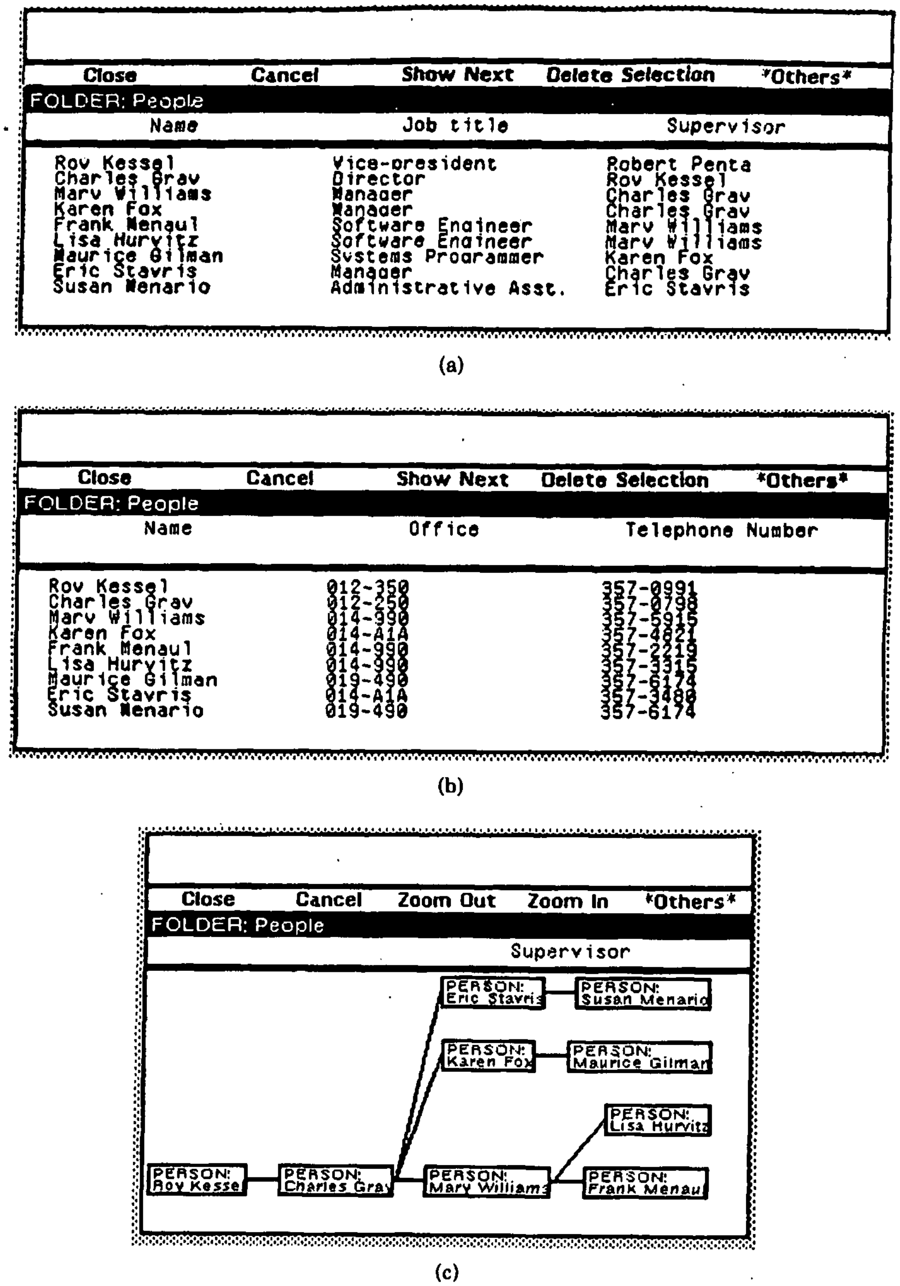

Fig. 4. Users can select which fields to display in tables that summarize a collection of objects.

added to a folder in two ways: (1) automatically, as the result of a rule action, or (2) manually using the `Add Link` action from the `*Others*` submenu on the folder. In both cases, the folders contain links to the objects, not the objects themselves. Therefore, the same object can appear in more than one folder. Other

actions for moving, copying, and deleting both objects and links are described in Section 3.7.

Object Lens currently provides two formats for displaying the contents of folders: *tables* and *trees*. Tables show the values of selected fields from the objects contained in the folder. For instance, Figure 4(a) shows a folder that contains objects representing people with the fields for a simple office directory displayed. Users can easily tailor the format of these displays by selecting from a menu the fields they want to have included in the table. For instance, Figure 4(b) shows the same folder, but with the display format changed to include a different set of fields.

Trees are graphs that show the objects in a folder and the links that connect these objects. Just as users can select the fields to be shown in a table, they can also select the fields from which links are shown. For instance, Figure 4(c) shows the same folder again, but this time in tree format with the "Supervisor" field selected as the one from which links are displayed. In this case, the display resembles a simple organization chart. In the current version of Object Lens, only the links in one field at a time can be displayed in a tree. In future versions, we plan to allow links from multiple fields to be shown with the links from different fields being displayed as different types of lines (e.g., solid, dotted).

When a new folder is created, the user is asked to select the default object type to be contained in the folder. The user is then allowed to choose from the fields of this default object type when selecting the fields to show in a table or when selecting the fields from which links are to be shown in a tree. Even though all folders have default object types, no strict type checking is enforced. If an object of an unexpected type is inserted into a folder, only the fields it shares with the default type are displayed in tables and trees.

3.7 Performing Actions on Objects

In addition to editing the contents of objects, users can also perform predefined actions on them. The actions that can be performed at any time depend on two primary factors: (1) the type of object being acted upon, and (2) the context in which the action is invoked.

3.7.1 *Object Specific Actions.* Each object type has a set of actions that can be performed on it. Some of these actions are "inherited" directly from the "parents" of the object type. Others may be modified or added specifically for this object type. For instance, there are some actions, such as `Hardcopy` and `Save`, that can be performed on all objects (i.e., all instances of `Thing` and all its subtypes). (Some of these actions, such as `Hardcopy`, are not yet implemented for all object types.) In addition, more specialized types of objects have other actions defined for them. For instance, agents have a `Run` action that triggers them to start running, and folders have a `Change Display Format` action that changes them from table format to tree format or vice versa.

In a few cases, the object specific actions depend, not just on the type of the object, but also on its state. For instance, messages created on the local workstation have a `Send` action, and messages received from elsewhere have actions such as `Answer` and `Forward`. So far these state-specific actions on objects are implemented as special cases. However, we would like to experiment with a more

general mechanism for representing state-specific actions and perhaps making this representation accessible to users. In some ways, this mechanism would be a generalization of the conversation manager in the Coordinator [21], which restricts the types of messages that a user can send at a given point in a conversation on the basis of the conversation state.

3.7.2 *Context Specific Actions.* There are some actions that can be applied to any kind of object but which can be invoked only from certain contexts. The primary contexts are (1) from an editor (like the one in Figure 1), (2) from a folder that contains the object, (3) from a rule operating on the object, and (4) from a link icon for the object.

For instance, when an object is being displayed in an editor, there are several kinds of actions, such as `Close`, `Move`, and `Shape`, that apply to the editing window. Other actions in an editor include (a) `Add Link` (insert at the current cursor position a link to another object selected by the user) and (b) `Cancel` (close the window without saving any of the changes made since the window was last opened).

When an object is displayed in a folder, other context-specific actions can be applied to it such as (a) `Show` (open an editor on the object) and (b) `Select` (select the item for some later folder action such as `Delete Selection`).

The actions that can be applied to an object by rules are discussed in Section 3.8. The actions that can be applied to link icons include `Show` (open an editor on the object) and `Delete` (delete this link to the object).

3.7.3 *Displaying and Invoking Actions.* Users invoke the above actions in slightly different ways depending on the context in which the object is displayed. If the object is displayed in an editor (like the one in Figure 1), then several of its most common actions are shown across the top of the editor, and all the other actions are shown in a menu that pops up when the `*Others*` action is selected.

When a link to an object is displayed (either as a link icon or as a row in a table), users can invoke actions in two ways. First, if users click on the link with both mouse buttons, a menu pops up showing all possible actions on the object. In addition, simply clicking on the link with the left mouse button invokes the most common action. For instance, clicking with the left button on a row in a table `Selects` the object for subsequent folder actions, whereas clicking with the left button on a link icon inside an editor `Shows` the object in another window on the screen.

3.8 Creating Agents and Rules

In some cases, agents can take actions automatically on behalf of their users. For instance, Figure 5 shows an example of a simple agent designed to help a user process incoming mail. When an agent is triggered, it applies a set of rules to a collection of objects in a folder. The agent in Figure 5 is applied to objects in the `New Mail` folder and is triggered by the arrival of new mail. That is, when mail is brought to the workstation, the mail program automatically inserts links to the new messages into the user's `New Mail` folder, and these `New Links` trigger the agent. In the current version of Object Lens, two other kinds of automatic triggers are available: `Daily at Midnight` and `On the Hour`.

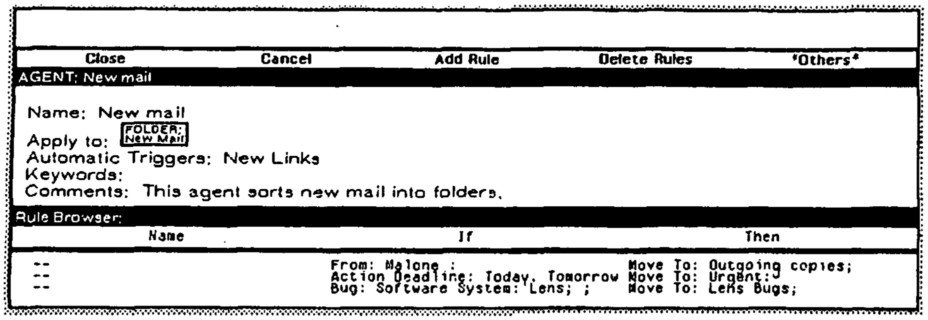

Fig. 5. Agents include a collection of rules and specifications for when and where to apply them.

The agent shown in Figure 5 includes several rules, one of which is shown in Figure 6. A rule contains an IF field (predicate) and a THEN field (action). Both parts of the rule contain links to other objects which are shown as embedded templates. The IF part of the rule is a *description*, a special kind of template that describes a set of instances in terms of the values of their fields. The THEN part of the rule is an Action object.

To construct the IF part of a rule, a user (a) clicks on the IF field with the middle mouse button, (b) selects "Descriptions" from the menu presented, and then (c) selects an object type from the tree of object types presented. This causes a description of the appropriate type to be inserted in the rule as an embedded template, and the user can then fill in the fields in this description to specify the values that must appear in particular fields for an object to satisfy the rule. As in the Information Lens, more complex specifications for a field can be constructed by combining strings with *and*, *or*, *not*, and parentheses (i.e., arbitrary Boolean combinations are possible within a field). If specifications appear in more than one field, then all specifications must be satisfied at once for the rule to succeed (i.e., specifications in different fields are implicitly *and*-ed). As in the other template-based editors in Object Lens, pop-up menus listing likely alternatives for a field are available in editing descriptions.

To specify the THEN part of a rule, a user simply clicks on the THEN field and selects an action from the menu of alternatives presented. These actions are applied to the *current object* (the object matched by the IF part of the rule) in the context of the *current folder* (the folder specified in the "Apply to" field of the agent). In some cases (such as the "Move" action shown here), the user also needs to fill in some fields in the embedded template for the action (e.g., the field specifying where the object is to be moved). The actions currently implemented in rules include the following: `copy` (add the current object to a different folder without removing it from the current folder), `move` (add the current object to a different folder and delete it from the current folder), `delete` (remove the object from the current folder), and `add keyword` (add the specified keyword to the Keywords field of the object). In addition, rules can invoke object specific actions, including the actions that apply to all objects such as `hardcopy` and `save`. We view the addition of more rule actions (and possibly the refinement of the rule syntax) as one of the important directions for our ongoing research.

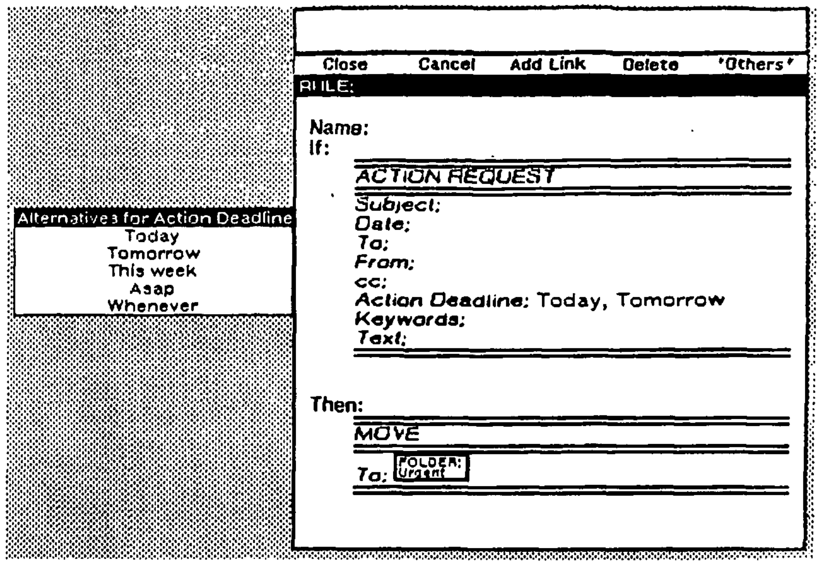

Fig. 6. Rules describe the objects that satisfy them and specify what action to perform on those objects.

The rules are applied in the order in which they appear in the agent's rule folder. Users can create extended reasoning chains by having some rules set characteristics of objects (using the `Add Keyword` action) which other rules test (by checking the `Keyword` field).

3.8.1 *Embedded Descriptions.* With the capabilities we have described so far, all rules must depend only on information contained in the objects to which they are being applied. For instance, a rule about a message can depend only on information contained in the message itself. It is often desirable, however, to be able to specify rules that also depend on other information contained elsewhere in the knowledge base. For instance, in the Information Lens system, if a user wanted to specify a rule that applied to all messages from vice presidents, the rule would have to include the names of all the vice presidents in the `From` field.

In Object Lens, it is possible to draw upon other information by having descriptions embedded within other descriptions. For instance, the rule shown in Figure 7 is satisfied if the message is from any person with a job title that includes "vice president." To apply this rule, the system checks to see whether the string in the `From` field of the message is the same as the `Name` of any `Person` object in the knowledge base that satisfies the description.

3.9 Navigating Through the System

The starting point for navigation through the Object Lens system is the Object Lens Icon, a window that shows whether the user has new mail waiting and includes a menu item to `Show Basics` (show the basic folders included in the system). The system folders accessible through the `Show Basics` action include (1) a folder containing all the other folders in the system, (2) a folder containing all the `Templates` defined in the system (Figure 3), (3) a folder containing all

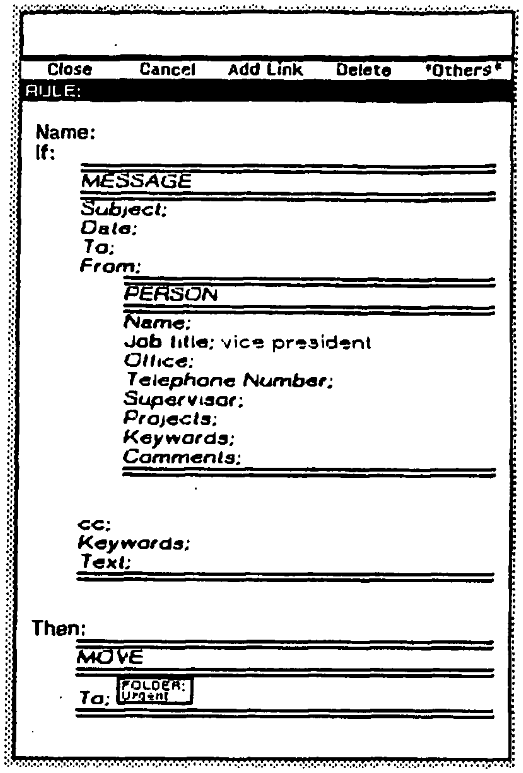

Fig. 7. Rules can use embedded descriptions to create complex queries.

the agents defined in the system, (4) a folder for each object type containing all the instances of that type in the system, and (5) the `New Mail` folder, into which new mail retrieved from the mail server is automatically inserted. In addition, we have designed but not fully implemented two other folders: (6) `Everything`, a virtual folder containing all objects in the system, and (7) `Orphans`, a virtual folder containing all objects to which no links exist.

These basic folders provide users with convenient starting points for locating any object in the system. In relatively small systems, users can browse through these folders directly. In larger systems, we expect users to let their agents search through the system folders to find objects that meet certain criteria. It is also possible for (a) individual users to create their own customized directory folders that contain the folders and other objects they most often use, and (b) application developers to create folders containing the objects used in their application.

3.10 Saving and Sharing Knowledge

One of the important research directions we plan to pursue in the Object Lens system involves different ways for people to save and share the kinds of knowledge described above. For instance, we are currently experimenting with linking Object Lens to a remote database server which contains large shared relational databases. This work is still at an early stage, but it is clear that the usefulness of Object Lens is significantly enhanced if it includes access to shared databases.

In the current version of Object Lens, we have preliminary solutions to the problems of saving and sharing knowledge that meet some, but not all, of the needs people have in this area.

3.10.1 *Saving Knowledge.* Users can save an object (or a collection of objects in a folder) at any time by performing the `Save` action on the object (or the folder). This action uses the file package commands from the underlying Loops and Lisp systems to store the objects in permanent files in a form that can be reloaded at any time. There is also a `Save` action on the main Object Lens icon that saves all the instances in the workstation.

The potential disadvantages of this approach to saving knowledge are that (1) it requires explicit user actions to save objects in permanent storage and (2) it requires all knowledge used by the system to be loaded onto the local workstation. Sharing remote databases, of course, helps solve these problems, but we expect that systems like Object Lens can be of value even without shared databases. For example, many users are already accustomed to explicitly saving their work in applications such as word processing, and even this task can be simplified by creating agents to run periodically (e.g., every night) and do automatic backups of selected objects.

3.10.2 *Sharing Knowledge by Sending Messages.* There are two ways users of Object Lens can share objects with each other: (1) by sending messages, and (2) by transferring files. In this subsection, we discuss sending messages; in the next, we discuss transferring files. When an Object Lens user sends a message, the message object is converted into text and sent via the existing mail system. Any connected electronic mail users can receive and read this textual message. When an Object Lens user receives the message, it is added as a new object in the receiver's knowledge base.

When a user sends a message containing an embedded object that is expanded (as in Figure 2), the embedded object is converted into (indented) text in the message in a form that (a) can be easily read by any receivers who are not using Object Lens, and (b) is reconverted into another embedded object when it is received by Object Lens users. When a user sends a message containing embedded objects that are *not* expanded (e.g., that are shown only as link icons), the names of the objects are included in the message in place of the link icons, but these names are not resolved back into link icons at the receiver's end.

One intriguing research direction here involves how to communicate embedded objects in such a way that they can be resolved into preexisting objects at the receiver's end. For example, if the sender's message contains a link to a person object, it would be nice for the receiver's system to be able to automatically resolve this link into the receiver's object representing the same person.

3.10.3 *Sharing Knowledge by Transferring Files.* The second way for users to share objects is by transferring files. As described above, it is easy for users to store on a file server the current state of a set of objects. Other users can then load these files to create (or update) the objects in their own workstations. Saving and loading these files can often be done automatically. For example, we expect that a common way for users to keep current versions of shared information such

as names, addresses, and job titles of people in their organization is to have someone maintain the official version of this information and periodically distribute updates to other users in the organization. Distributing these updates could be done in several ways: (1) the maintainer could have automatic agents that periodically store the current versions on a file server, and the other users could have automatic agents that periodically load the most recent versions, or (2) the maintainer could explicitly send out messages announcing the availability of files containing updated objects, and the other users could have agents that automatically load the files announced in such messages (e.g., a rule might load all files specified in "Official file update" messages from the official maintainer).

One potential problem with this approach is that any changes users have made to their local copies of objects (e.g., any notes they had added in the Comments field) are lost when a new version of the object is loaded. To help solve this problem, we are currently investigating more specialized updating actions for agents to use. With this approach, the official maintainer will be able to distribute update messages that specify changes in particular fields of particular objects. Users can then set up agents that make these updates automatically under most conditions, but under certain conditions the user might be notified before the update is made (e.g., if the field about to be modified has previously been changed by the user). In some cases, the user might want to have the change made automatically but also want to be notified (e.g., if someone in the user's group is changing phone numbers).

4. OTHER APPLICATIONS

In this section, we give more examples of how the above features can be combined to create a variety of cooperative work applications.

4.1 Task Tracking

One frequently mentioned capability for cooperative work applications is the ability to keep track of the tasks people are supposed to do (e.g, [18, 22]). For instance, such systems can help answer questions like: What tasks have other people requested me to do? Are any of these tasks overdue? What tasks have I requested other people to do for me?

Supporting capabilities like this in Object Lens is a straightforward matter. For instance, the system already includes message types for action requests and commitments. Even in the Information Lens, it was possible to automatically sort these messages into folders according to who is to perform the task, which project it involves, and so forth. In the Information Lens, however, the summary display of a folder's contents shows only the standard message header fields: `From`, `Date`, and `Subject`. To see more about the tasks, individual messages have to be displayed, one at a time. In Object Lens, the messages within a folder can easily be summarized by displaying whatever fields the user chooses. For example, Figure 8 shows a table display of action request messages that includes the action deadline.

4.2 Intelligent Message Sorting: Engineering Change Notices

As we have described in more detail elsewhere [15], an intriguing example of a cooperative work problem involves disseminating information about changes in

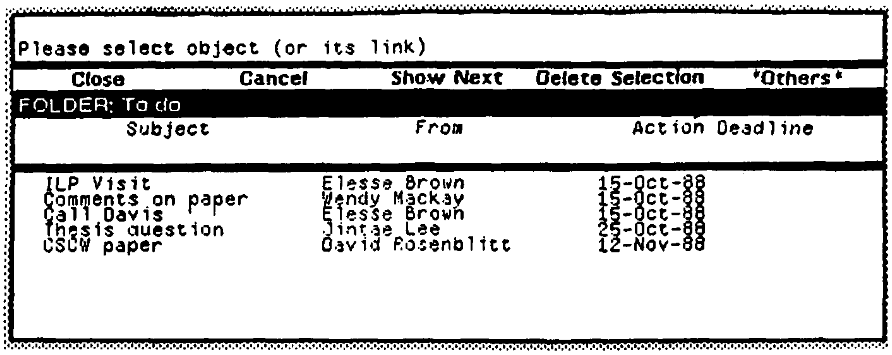

Fig. 8. Tables can be used to summarize selected fields from Action Request messages.

product specifications (often called "engineering change notices") to the appropriate people in an organization. It was already possible in the Information Lens to sort engineering change notices according to the contents of fields such as `Part Affected`, `Type of Change`, and `Severity`. In Object Lens, it is possible to use additional knowledge to do even more intelligent sorting. For instance, Figure 9 shows a rule that uses a doubly embedded description to select all change notices that involve parts for which anyone reporting to a particular manager is responsible.

4.3 Database Retrieval

There are clearly many cases in both individual and cooperative work when it is useful to be able to automatically retrieve objects that satisfy certain conditions from a database. Object Lens provides a simple way to perform database queries: Users can simply create agents that scan the objects in one folder and insert links to selected objects into another folder. The rules in the agents specify the criteria for selecting objects.

For instance, suppose you wanted to find all the technical staff members who were assigned to both the project code-named "Dragon" and the one code-named "Lancelot." Figure 10 shows a rule that would retrieve all such people. Instead of listing all the technical job titles by name ("software engineer," "systems programmer," etc.), the rule includes an embedded description to determine whether a particular job title is on the technical, as opposed to the managerial or administrative, career ladder.

In addition to this general interface for database retrieval, we have also implemented a specialized feature in Object Lens for determining the recipients of messages. With this feature, descriptions (like that shown in the IF field of Figure 10) can be embedded in the `To` and `cc` fields of a message. Then, when the message is sent, these descriptions are automatically applied to all the Person objects in the local knowledge base, and the resulting people are inserted in the `To` and `cc` fields. This feature allows senders to create distribution lists that are dynamically computed at message-sending time on the basis of current information about people in their database (see [23] for a similar capability).

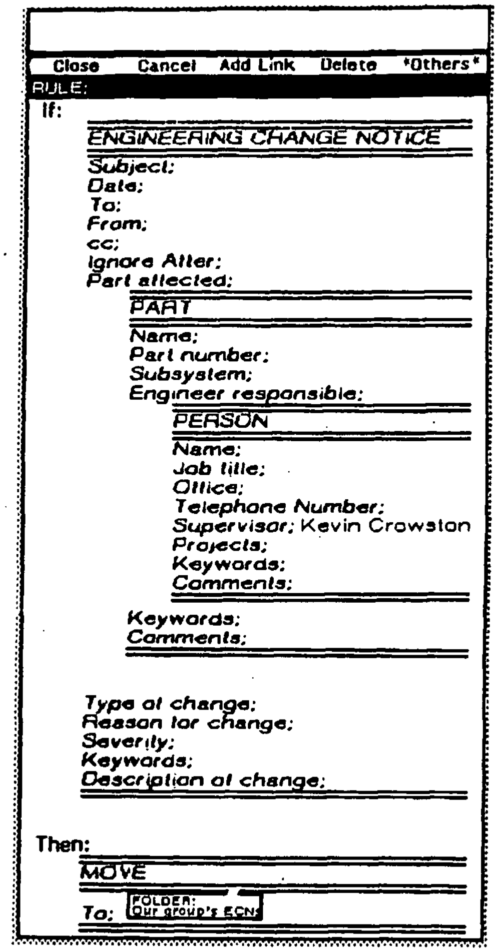

Fig. 9. Rules can include multiple levels of embedded descriptions that refer to linked objects throughout the knowledge base.

4.4 Hypertext

As noted above, it is a straightforward matter to use many of the features of a hypertext system in Object Lens (e.g., [3, 6, 9]). For instance, our system currently contains an object type called Text that displays only two fields: Name and Text. The Text field of a Text object can contain links to as many other objects as desired. For example, Figure 11 shows a sample Text object that contains links to people and bibliographic citations as well as to another Text object.

In addition to the usual benefits of hypertext systems, Object Lens derives additional benefits from its integration of hypertext with other database, messaging, and computational capabilities. For instance, in order to insert a link to another node in a hypertext system, a user must first find the node to which the link will be made. In Object Lens, the database retrieval capabilities described above can be used to automatically find objects (such as people or bibliographic citations) that satisfy certain criteria. Then links to these objects can be inserted into the text. One desirable feature found in some hypertext systems that is not yet included in Object Lens is the ability to show and follow the incoming links

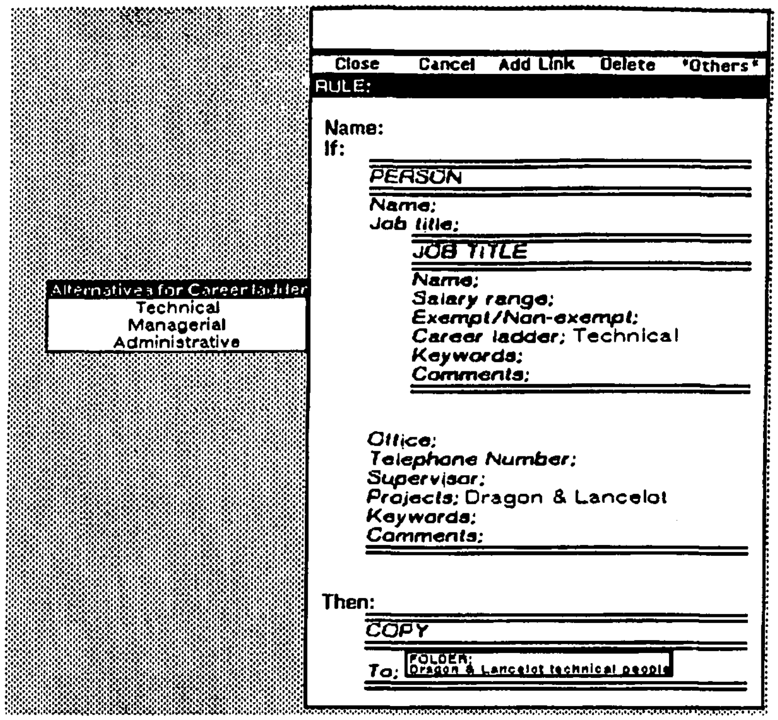

Fig. 10. Agents can retrieve all the objects from a database that satisfy certain criteria.

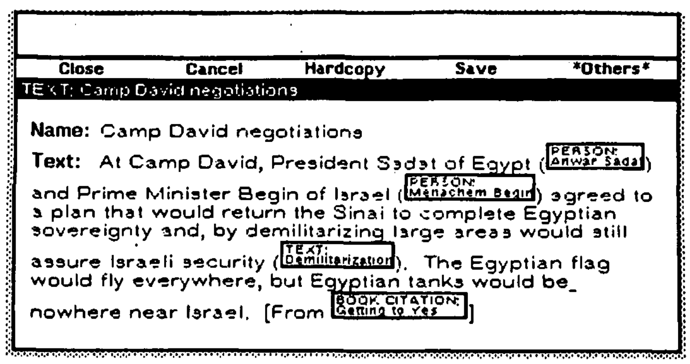

Fig. 11. Hypertext documents can include links, not only to other text passages, but also to other object types such as people and bibliographic citations.

to an object. We would like to implement this capability as another action available on all objects.

Even though the relationship between Object Lens and previous hypertext systems is not the primary focus of this paper, it is interesting to observe that Object Lens appears to have some functionality in at least four of the seven areas

that Halasz [8] listed as being needed in the next generation of hypermedia systems (search and query, computational engines, collaborative work, and tailorability).

5. CONCLUSION

In this paper, we have described a system called Object Lens that integrates facilities for hypertext, object-oriented databases, electronic messaging, and rule-based agents. Using the basic primitives provided by this system, we believe it is relatively easy to create a wide variety of cooperative work applications. We have shown several such applications here, and an important focus of our ongoing research will be to test the generality of the framework further by implementing more applications within it.

Object Lens is an example of a semiformal system, a system that represents knowledge in a way that both people and their computational agents can process intelligently. We believe that much of the power and flexibility of this system results from its choice of primitives (semistructured objects, customizable folders, and semiautonomous agents) and from the template-based interfaces that make these primitives both visible and changeable by inexperienced computer users.

ACKNOWLEDGMENTS

We would especially like to thank Ken Grant who suggested some of the earliest ideas that led to Object Lens and Jin Lee who helped debug the most recent version. The Object Lens system and this paper have also benefited from conversations with Cheryl Clark, Kevin Crowston, Randy Davis, Frank Halasz, Mitch Kapor, Stan Lanning, Wendy Mackay, Ramana Rao, Randy Trigg, David Rosenblitt, and Franklyn Turbak.

REFERENCES

1. CONKLIN, J. Hypertext: An introduction and survey. *IEEE Computer 20*, 9 (1987), 17–41.
2. CROWSTON, K., AND MALONE, T. W. Computational agents to support cooperative work. Working Paper No. 2008-88, Center for Information Systems Research, Massachusetts Institute of Technology, Cambridge, Mass., 1988.
3. DELISLE, N., AND SCHWARTZ, M. Contexts—a partitioning concept for hypertext. *ACM Trans. Off. Inf. Syst. 5*, 2 (Apr. 1987), 168–186.
4. DITTRICH, D., AND DAYAL, U., Eds. In *Proceedings of the International Workshop on Object-Oriented Database Systems* (Asilomar, Calif., Sept. 23–26). IEEE Computer Society, Washington, D.C., 1986.
5. FIKES, R., AND KEHLER, T. The role of frame-based representation in reasoning. *Commun. ACM 28*, 9 (Sept. 1985), 904.
6. GARRETT, L. N., SMITH, K. E., AND MEYROWITZ, N. Intermedia: Issues, strategies, and tactics in the design of a hypermedia document system. In *Proceedings of the Conference on Computer-Supported Cooperative Work* (Austin, Tex., Dec. 3–5). ACM, New York, 1986, 163–174.
7. GREIF, I. Computer-supported cooperative work: Breakthroughs for user acceptance (Panel description). In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '88)* (Washington, D.C., May 16–19). ACM, New York, 1988, pp. 113–114.
8. HALASZ, F. G. Reflections on NoteCards: Seven issues for the next generation of hypermedia systems. *Commun. ACM 31*, 7 (July 1987), 836–855.
9. HALASZ, F. G., MORAN, T. P., AND TRIGG, R. H. NoteCards in a nutshell. In *Proceedings of the 1987 ACM Conference on Human Factors in Computer Systems (CHI + GI '87)* (Toronto, Ontario, Apr. 5–9). ACM, New York, 45–52.

10. LAI, K. Y. Essays on Object Lens: A tool for supporting information sharing. Master's thesis, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Mass., 1987.
11. LEE, J., AND MALONE, T. W. How can groups communicate when they use different languages? Translating between partially shared type hierarchies. In *Proceedings of the ACM Conference on Office Information Systems* (Palo Alto, Calif., Mar. 23-25). ACM, New York, 1988, pp. 22-29.
12. LEE, J., AND MALONE, T. W. Partially shared views: A scheme for communicating among groups that use different type hierarchies. Sloan School of Management Working Paper, Massachusetts Institute of Technology, Cambridge, Mass., Sept., 1988.
13. MALONE, T. W., GRANT, K. R., LAI, K.-Y., RAO, R., AND ROSENBLITT, D. Semistructured messages are surprisingly useful for computer-supported coordination. *ACM Trans. Off. Syst. 5*, 2 (Apr. 1987), 115-131.
14. MALONE, T. W., GRANT, K. R., TURBAK, F. A., BROBST, S. A., AND COHEN, M. D. Intelligent information-sharing systems. *Commun. ACM 30*, 5 (May 1987), 390-402.
15. MALONE, T. W., GRANT, K. R., LAI, K.-Y., RAO, R., AND ROSENBLITT, D. The Information Lens: An intelligent system for information sharing and coordination. In *Technological Support for Work Group Collaboration*, M. H. Olson, Ed. Lawrence Erlbaum, Hillsdale, N.J., 1989.
16. SHRIVER, B., AND WEGNER, P. *Research Directions in Object-Oriented Programming*. MIT Press, Cambridge, Mass., 1987.
17. STEFIK, M., AND BOBROW, D. G. Object-oriented programming: Themes and variations. *AI Magazine* (Spring 1986), 40-62.
18. SLUIZER, S., AND CASHMAN, P. M. XCP: An experimental tool for supporting office procedures. In *IEEE 1984 Proceedings of the 1st International Conference on Office Automation* (Silver Spring, Md.). IEEE Computer Society, Washington, D.C., 1984, pp. 73-80.
19. TOU, F. N., WILLIAMS, M. D., FIKES, R. E., HENDERSON, D. A., AND MALONE, T. W. RABBIT: An intelligent database assistant. In *Proceedings of the National Conference of the American Association for Artificial Intelligence* (Pittsburgh, Pa., Aug. 18-20). American Association for Artificial Intelligence, Philadelphia, Pa., 1982, pp. 314-318.
20. TURBAK, F. A. *Grasp*: A visible and manipulable model for procedural programs. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Mass., 1986.
21. WINOGRAD, T. A language/action perspective on the design of cooperative work. *Human Computer Interaction 3*, 1 (1988), 3-30.
22. WINOGRAD, T., AND FLORES, F. *Understanding Computers and Cognition: A New Foundation For Design*. Ablex, Norwood, NJ, 1986.
23. ZLOOF, M. M. QBE/OBE: A language for office and business automation. *IEEE Computer 14*, 5 (May 1981), 13-22.

Received June 1988; revised October 1988; accepted October 1988

E4068-01
(米)

COMBINED DECLARATION AND POWER OF ATTORNEY
(宣誓書及び委任状)

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name, I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

"METHOD AND APPARATUS FOR SEARCHING AND DISPLAYING STRUCTURED DOCUMENT"

the specification of which: (check one) [X] is attached hereto.

[ ] was filed on ____________
as Application Serial No. ____________
and was amended on ____________
(if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended, by any amendment referred to above.

I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known to me which is material to patentability in accordance with Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date earlier than that of the application(s) on which priority is claimed:

Prior Foreign Application(s)

(Number)	(Country)	(Day/Month/Year Filed)	Priority Claimed Yes	Priority Claimed No
09-190716	Japan	1 July, 1997	[X]	[ ]
09-195408	Japan	22 July, 1997	[X]	[ ]
			[ ]	[ ]

I hereby claim the benefit under Title 35, United States Code, 120 of any United States application(s) or PCT international application(s) designating the United States of America that is/are listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in that/those prior application(s) in the manner provided by the first paragraph of Title 35, United States Code § 112, I acknowledge the duty to discolse to the United States Patent and Trademark Office all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)

(Continued on Page 2)

Object lens: a "spreadsheet" for cooperative work

Full text Pdf (1.78 MB)

Source ACM Transactions on Information Systems (TOIS) archive
Volume 6 , Issue 4 (October 1988) table of contents
Pages: 332 - 353
Year of Publication: 1988
ISSN:1046-8188

Authors Kum-Yew Lai Mssachusetts Institute of Technology, Cambridge
Thomas W. Malone Massachusetts Institute of Technology, Cambridge
Keh-Chiang Yu Massachusetts Institute of Technology, Cambridge

Publisher ACM Press New York, NY, USA

DOI Bookmark: Use this link to bookmark this Article: http://doi.acm.org/10.1145/58566.59298
What is a DOI?

↑ ABSTRACT

Object Lens allows unsophisticated computer users to create their own cooperative work applications using a set of simple, but powerful, building blocks. By defining and modifying templates for various semistructured objects, users can represent information about people, tasks, products, messages, and many other kinds of information in a form that can be processed intelligently by both people and their computers. By collecting these objects in customizable folders, users can create their own displays which summarize selected information from the objects in table or tree formats. Finally, by creating semiautonomous agents, users can specify rules for automatically processing this information in different ways at different times. The combination of these primitives provides a single consistent interface that integrates facilities for object-oriented databases, hypertext, electronic messaging, and rule-based intelligent agents. To illustrate the power of this combined approach, we describe several simple examples of applications (such as task tracking, intelligent message routing, and database retrieval) that we have developed in this framework.

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 Jeff Conklin, Hypertext: an introduction and survey, Computer, v.20 n.9, p.17-41, Sept. 1987

2 CROWSTON, K., AND MALONE, T. W. Computational agents to support cooperative work. Working Paper No. 2008-88, Center for Information Systems Research, Massachusetts Institute of

Technology, Cambridge, Mass., 1988.

3 Norman M. Delisle , Mayer D. Schwartz, Contexts—a partitioning concept for hypertext, ACM Transactions on Information Systems (TOIS), v.5 n.2, p.168-186, April 1987

4 Proceedings on the 1986 international workshop on Object-oriented database systems, September 1986

5 Richard Fikes , Tom Kehler, The role of frame-based representation in reasoning, Communications of the ACM, v.28 n.9, p.904-920, Sept. 1985

6 L. Nancy Garrett , Karen E. Smith , Norman Meyrowitz, Intermedia: issues, strategies, and tactics in the design of a hypermedia document system, Proceedings of the 1986 ACM conference on Computer-supported cooperative work, December 03-05, 1986, Austin, Texas

7 I. Greif , J. S. Brown , E. Dyson , M. Kapor , T. Malone, Computer-supported cooperative work: breakthroughs for user acceptance, Proceedings of the SIGCHI conference on Human factors in computing systems, p.113-114, May 15-19, 1988, Washington, D.C., United States

8 Frank,G. Halasz, Reflections on NoteCards: seven issues for the next generation of hypermedia systems, Communications of the ACM, v.31 n.7, p.836-852, July 1988

9 Frank G. Halasz , Thomas P. Moran , Randall H. Trigg, Notecards in a nutshell, Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface, p.45-52, April 05-09, 1987, Toronto, Ontario, Canada

10 LAI, K.Y. Essays on Object Lens: A tool for supporting information sharing. Master's thesis, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Mass., 1987.

11 Jintae Lee , Thomas W. Malone, How can groups communicate when they use different languages?, Proceedings of the ACM SIGOIS and IEEECS TC-OA 1988 conference on Office information systems, p.22-29, March 23-25, 1988, Palo Alto, California, United States

12 LEE, J., AND MALONE, T. W. Partially shared views: A scheme for communicating among groups that use different type hierarchies. Sloan School of Management Working Paper, Massachusetts Institute of Technology, Cambridge, Mass., Sept., 1988.

13 Thomas W. Malone , Kenneth R. Grant , Kum-Yew Lai , Ramana Rao , David Rosenblitt, Semistructured messages are surprisingly useful for computer-supported coordination, ACM Transactions on Information Systems (TOIS), v.5 n.2, p.115-131, April 1987

14 Thomas W Malone , Kenneth R Grant , Franklyn A Turbak , Stephen A Brobst , Michael D Cohen, Intelligent information-sharing systems, Communications of the ACM, v.30 n.5, p.390-402, May 1987

15 Thomas W. Malone , Kenneth R. Grant , Kum-Yew Lai , Ramana Rao , David A. Rosenblitt, The information lens: an intelligent system for information sharing and coordination, Technological support for work group collaboration, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 1989

16 Bruce Shriver , Peter Wegner, Research directions in object-oriented programming, MIT Press, Cambridge, MA, 1987

17 Mark Stefik , Daniel Bobrow, Object-oriented programming: Themes and variations, AI Magazine, v.6 n.4, p.40-62, Winter 1986

18 SLUIZER, S., AND CASHMAN, P. M. XCP: An experimental tool for supporting office procedures.

In IEEE 1984 Proceedings of the 1st International Conference on Office Automation (Silver Spring, Md.). IEEE Computer Society, Washington, D.C., 1984, pp. 73-80.

19 TOU, F. N., WILLIAMS, M. D., F~KES, R. E., HENDERSON, D. A., AND MALONE, T.W. RABBIT: An intelligent database assistant, in Proceedings of the National Conference of the American Association/or Artificial Intelligence {Pittsburgh, Pa., Aug. 18-20). American Association for Artificial Intelligence, Philadelphia, Pa., 1982, pp. 314-318.

20 TuRsAK, F. A. Grasp: A visible and manipulable model for procedural programs. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Mass., 1986.

21 WINOGRA{, T. A language/action perspective on the design of cooperative work. Human Computer Interaction 3, 1 (1988), 3-30.

22 Terry Winograd , Fernando Flores, Understanding computers and cognition, Ablex Publishing Corp., Norwood, NJ, 1986

23 ZLOOF, M.M. QBE/OBE: A language for office and business automation. IEEE Computer 14, 5 (May 1981), 13-22.

↑ CITINGS 35

Kum-Yeq Lai , Thomas W. Malone, Object lens: letting end-users create cooperative work applications, Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology, p.425-426, April 27-May 02, 1991, New Orleans, Louisiana, United States

Gary Perlman, Information retrieval techniques for hypertext in the semi-structured toolkit, Proceedings of the fifth ACM conference on Hypertext, p.260-267, November 14-18, 1993, Seattle, Washington, United States

Gio Wiederhold, Mediators in the Architecture of Future Information Systems, Computer, v.25 n.3, p.38-49, March 1992

Jintae Lee, Extending the Potts and Bruns model for recording design rationale, Proceedings of the 13th international conference on Software engineering, p.114-125, May 13-17, 1991, Austin, Texas, United States

Larry Press, Personal computing: personal computers as research tools, Communications of the ACM, v.34 n.4, p.19-25, April 1991

Jintae Lee, SIBYL: a tool for managing group design rationale, Proceedings of the 1990 ACM conference on Computer-supported cooperative work, p.79-92, October 07-10, 1990, Los Angeles, California, United States

Allen E. Milewski , Thomas M. Smith, An experimental system for transactional messaging, Proceedings of the international ACM SIGGROUP conference on Supporting group work: the integration challenge, p.325-330, November 16-19, 1997, Phoenix, Arizona, United States

Paul Resnick , Robert A. Virzi, Skip and scan: cleaning up telephone interface, Proceedings of the SIGCHI conference on Human factors in computing systems, p.419-426, May 03-07, 1992, Monterey, California, United States

Paul Resnick, HyperVoice: a phone-based CSCW platform, Proceedings of the 1992 ACM conference on Computer-supported cooperative work, p.218-225, November 01-04, 1992, Toronto, Ontario, Canada

Thomas W. Malone , Kum-Yew Lai , Christopher Fry, Experiments with Oval: a radically tailorable tool for cooperative work, Proceedings of the 1992 ACM conference on Computer-supported cooperative work, p.289-297, November 01-04, 1992, Toronto, Ontario, Canada

Jintae Lee , Thomas W. Malone, Partially shared views: a scheme for communicating among groups that use different type hierarchies, ACM Transactions on Information Systems (TOIS), v.8 n.1, p.1-26, Jan. 1990

Christine M. Neuwirth , James H. Morris , Susan Harkness Regli , Ravinder Chandhok , Geoffrey C. Wenger, Envisioning communication: task-tailorable representations of communication in asynchronous work, Proceedings of the 1998 ACM conference on Computer supported cooperative work, p.265-274, November 14-18, 1998, Seattle, Washington, United States

Paul Resnick, Phone-based CSCW: tools and trials, ACM Transactions on Information Systems (TOIS), v.11 n.4, p.401-424, Oct. 1993

Pattie Maes, Agents that reduce work and information overload, Communications of the ACM, v.37 n.7, p.30-40, July 1994

Eric K. McCall , Lori A. Clarke , Leon J. Osterweil, An adaptable generation approach to agenda management, Proceedings of the 20th international conference on Software engineering, p.282-291, April 19-25, 1998, Kyoto, Japan

Clarence Martens , Frederick H. Lochovsky, OASIS: a programming environment for implementing distributed organizational support systems, ACM SIGOIS Bulletin, v.12 n.2-3, p.29-42, Nov. 1991

Rajan M. Lukose , Eytan Adar , Joshua R. Tyler , Caesar Sengupta, SHOCK: communicating with computational messages and automatic private profiles, Proceedings of the 12th international conference on World Wide Web, May 20-24, 2003, Budapest, Hungary

Jakob Nielsen, Noncommand user interfaces, Communications of the ACM, v.36 n.4, p.83-99, April 1993

David A. Marca, Augmenting SADT to develop computer support for cooperative work, Proceedings of the 13th international conference on Software engineering, p.94-103, May 13-17, 1991, Austin, Texas, United States

Mark S. Ackerman, Augmenting the organizational memory: a field study of answer garden, Proceedings of the 1994 ACM conference on Computer supported cooperative work, p.243-252, October 22-26, 1994, Chapel Hill, North Carolina, United States

Peter G. Anick , Rex A. Flynn , David R. Hanssen, Addressing the requirements of a dynamic corporate textual information base, Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval, p.163-172, October 13-16, 1991, Chicago, Illinois, United States

Heikki Hämmäinen , Eero Eloranta , Jari Alasuvanto, Distributed form management, ACM Transactions on Information Systems (TOIS), v.8 n.1, p.50-76, Jan. 1990

Prasanta Bose , Xiaoqing Zhou, WWAC: WinWin abstraction based decision coordination, ACM SIGSOFT Software Engineering Notes, v.24 n.2, p.127-136, March 1999

Barry Boehm , Prasanta Bose , Ellis Horowitz , Ming June Lee, Software requirements negotiation and renegotiation aids, Proceedings of the 17th international conference on Software engineering, p.243-253, April 24-28, 1995, Seattle, Washington, United States

Jerry Fowler , Donald G. Baker , Ross Dargahi , Vram Kouramajian , Hillary Gilson , Kevin Brook Long , Cynthia Petermann , G. Anthony Gorry, Experience with the virtual notebook system: abstraction in hypertext, Proceedings of the 1994 ACM conference on Computer supported cooperative work, p.133-143, October 22-26, 1994, Chapel Hill, North Carolina, United States

Murugappan Palaniappan , Nicole Yankelovich , George Fitzmaurice , Anne Loomis , Bernard Haan , James Coombs , Norman Meyrowitz, The envoy framework: an open architecture for agents, ACM Transactions on Information Systems (TOIS), v.10 n.3, p.233-264, July 1992

Thomas W. Malone , Kum-Yew Lai , Christopher Fry, Experiments with Oval: a radically tailorable tool for cooperative work, ACM Transactions on Information Systems (TOIS), v.13 n.2, p.177-205, April 1995

Thomas W. Malone , Kevin Crowston, What is coordination theory and how can it help design cooperative work systems?, Proceedings of the 1990 ACM conference on Computer-supported cooperative work, p.357-370, October 07-10, 1990, Los Angeles, California, United States

Laura Teodosio , Walter Bender, Salient stills, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP), v.1 n.1, p.16-36, February 2005

Franca Garzotto , Paolo Paolini , Daniel Schwabe, HDM—a model-based approach to hypertext application design, ACM Transactions on Information Systems (TOIS), v.11 n.1, p.1-26, Jan. 1993

J. Alfredo Sánchez , John J. Leggett , John L. Schnase, HyperActive: extending an open hypermedia architecture to support agency, ACM Transactions on Computer-Human Interaction (TOCHI), v.1 n.4, p.357-382, Dec. 1994

Mark Bernstein , Jay David Bolter , Michael Joyce , Elli Mylonas, Architectures for volatile hypertext, Proceedings of the third annual ACM conference on Hypertext, p.243-260, December 15-18, 1991, San Antonio, Texas, United States

Thomas W. Malone , Kevin Crowston, The interdisciplinary study of coordination, ACM Computing Surveys (CSUR), v.26 n.1, p.87-119, March 1994

Gary Marchionini , John Sibert, An agenda for human-computer interaction: science and engineering serving human needs, ACM SIGCHI Bulletin, v.23 n.4, p.17-32, Oct. 1991

↑ INDEX TERMS

Primary Classification:
H. Information Systems
↪ **H.1** MODELS AND PRINCIPLES

Additional Classification:
H. Information Systems
↪ **H.2** DATABASE MANAGEMENT
↪ **H.2.1** Logical Design
↪ **Subjects:** Schema and subschema; Data models

↪ **H.2.3** Languages
↪ **Subjects:** Data description languages (DDL)
↪ **H.2.4** Systems
↪ **Subjects:** Distributed databases
↪ **H.3** INFORMATION STORAGE AND RETRIEVAL
↪ **H.4** INFORMATION SYSTEMS APPLICATIONS

I. Computing Methodologies
↪ **I.2** ARTIFICIAL INTELLIGENCE
↪ **I.2.4** Knowledge Representation Formalisms and Methods
↪ **Subjects:** Frames and scripts; Predicate logic

General Terms:
Design, Human Factors

↑ REVIEW

"Joseph L. Podolsky"

The use of objects has become the latest fad in software engineering, picking up where artificial intelligence left off. The description of the Object Lens in this paper is useful, however, because it provides some specific examples of how objects might work in a real-world environment and offers the notion of semiautomated, semiautonomous software behavior. The Object Lens is a user interface that integrates hypertext, object-oriented databases, electronic messaging, and rule-based intelligent agents. Still in the prototype stage, this system succeeds the earlier, less capable Information Lens and is designed to be a knowledge-based environment for developing applications in a cooperative fashion, where the work is distributed among several people, teams, or geographic locations. The authors try to relate their system to commercially successful spreadsheet packages by offering a very general definition of spreadsheet software. The success of products like Lotus 1-2-3, however, was largely due to a factor they do not mention—the target market's familiarity with the paper spreadsheet. Although the Object Lens does incorporate elements that have analogies in nonautomated processes, the authors do not state these analogies explicitly, and the system has no single manual metaphor. It does, however, resemble a meeting where project team members exchange various finished pieces of programs and documentation that other team members then use. The authors call their system "semiformal," that is, one that represents and automatically processes certain information in formally specified ways, represents and facilitates the human processing of the same or different information in ways that are not formally specified, and allows the boundary between formal processing by the system and informal processing by humans to be changed easily and frequently. Their system can also represent and manipulate "semistructured" objects; fields can be structured or not, and the data in them can be restricted or flexible in type or content. The Object Lens uses "template-based" user interfaces, and displays may be defined by the user. As in other object-oriented software, all elements of the software can be treated as objects to be called by or embedded into other objects. New objects can inherit some or all of the characteristics of the old objects, and objects can be collected into folders and linked into tables or tree structures. Semiautonomous agents can process information. These agents can take a series of actions without human intervention; the processing rules, however, are easily visible to human users, who can change them. The agents may also refer objects to a human user for action. Agents are also objects, of course, that can be called by or embedded in other objects with full or partial inheritance. Frankly, this paper did not add much to my knowledge of objects, but the notion of human/computer interaction that the various "semi-" functions offer did impress me. I know too much about computers and software to trust them very much, and I like the idea that the power of

this system is visible to and under the convenient and explicit control of the human user. The Object Lens is currently a prototype implemented in Interlisp-D on Xerox 1100 workstations. The authors claim that all the features they describe are actually implemented, although some are not fully tested. This good work in progress may eventually result in a powerful family of products, but even if the Object Lens should fail in the marketplace, the ideas in this paper are valuable and will further research in the area by ratcheting our expectations a few notches higher. Online Computing Reviews Service

Collaborative Colleagues:

Kum-Yew Lai:
- Christopher Fry
- Kenneth R. Grant
- Jintae Lee
- Thomas W. Malone
- Ramana Rao
- David Rosenblitt
- David A. Rosenblitt
- Keh-Chiang Yu

Thomas W. Malone:
- Mark S. Ackerman
- Robert I. Benjamin
- Abraham Bernstein
- Erik Brynjolfsson
- Paul M. Cashman
- Kevin Crowston
- Kevin Ghen Crowston
- Bill Curtis
- Chrysanthos Dellarocas
- Chrysanthos Nicholas Dellarocas
- Christopher Fry
- Kenneth R. Grant
- Irene Greif
- Vijay Gurbaxani
- George Herman
- George A. Herman
- Anatol Holt
- Ajit Kambil
- Mark Klein
- Herb Krasner
- Kum-Yeq Lai
- Kum-Yew Lai
- Robert Laubacher
- Robert J. Laubacher
- Jintae Lee
- James Levin
- F. Lin
- Felix Lin
- Fred L. Luconi
- Michael S. Scott Morton
- Elissa O'Donnell
- Charles S. Osborn
- Brian Pentland
- John Quimby
- Ramana Rao
- Paul Resnick
- John F. Rockart
- David Rosenblitt
- David A. Rosenblitt
- Michael S. ScottMorton
- Ben Shneiderman
- F. Don Smith
- John B. Smith
- Stephen A. Smith
- Ronald Stamper
- Terry Winograd
- George Wyner
- George Michael Wyner
- Joanne Yates
- Keh-Chiang Yu

Keh-Chiang Yu:
- Kum-Yew Lai
- Thomas W. Malone
- Lawrence C. N. Tseung

Peer to Peer - Readers of this Article have also read:

- M^4: a metamodel for data preprocessing **Proceedings of the 4th ACM international workshop on Data warehousing and OLAP**
  Anca Vaduva , Jörg-Uwe Kietz , Regina Zücker

- Inferring constraints from multiple snapshots **ACM Transactions on Graphics (TOG)** 12, 4
  David Kurlander , Steven Feiner

- Data structures for quadtree approximation and compression **Communications of the ACM** 28, 9
  Hanan Samet

- A hierarchical single-key-lock access control using the Chinese remainder theorem **Proceedings of the 1992 ACM/SIGAPP Symposium on Applied computing**
  Kim S. Lee , Huizhu Lu , D. D. Fisher

- Putting innovation to work: adoption strategies for multimedia communication systems

Communications of the ACM 34, 12
Ellen Francik , Susan Ehrlich Rudman , Donna Cooper , Stephen Levine